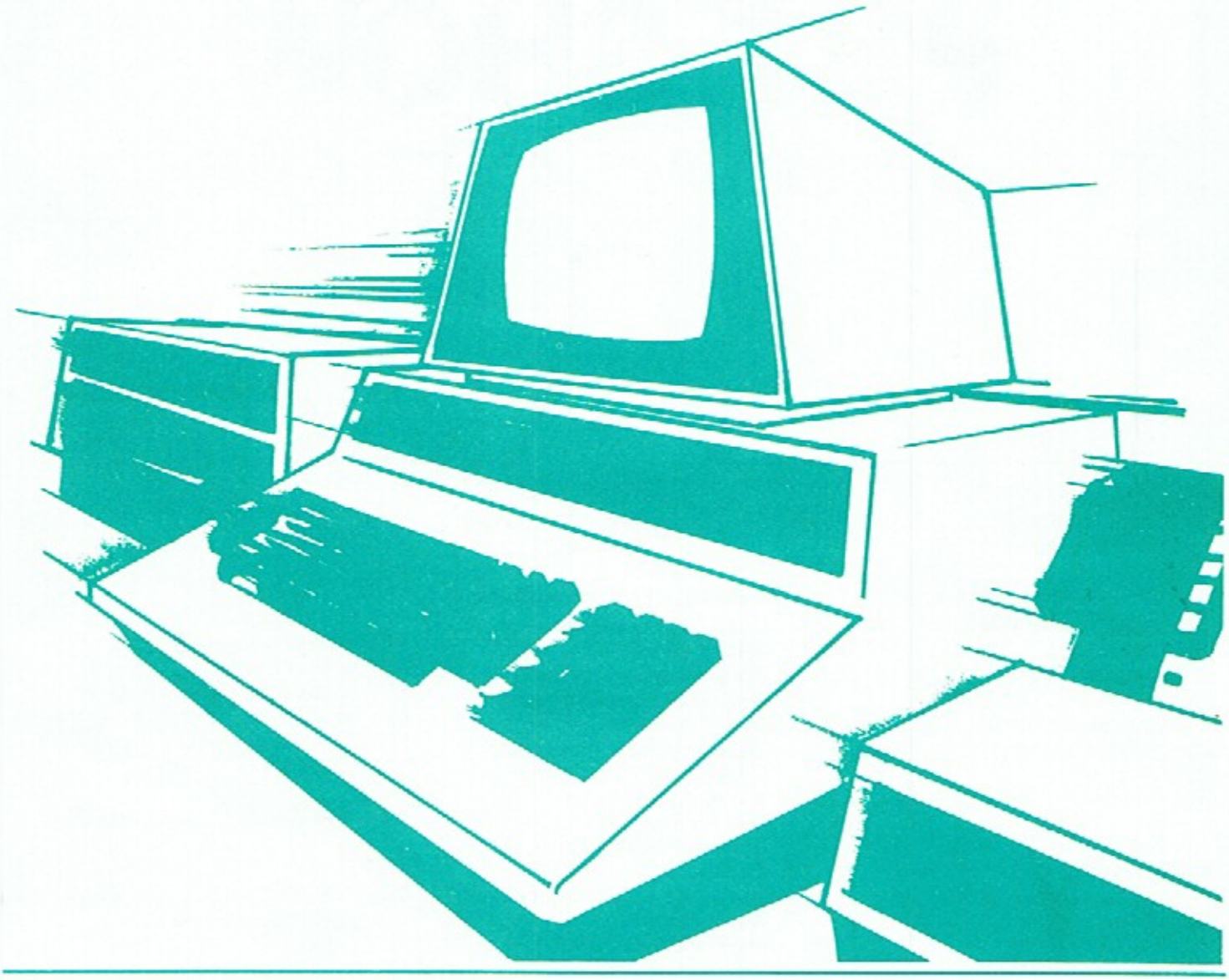
CPUCN

The Official Commodore Pet Users Club Newsletter



Volume 2

Issue 5

commodore

Contents

Commodore News
Teach In
Microcomputers and the Businessman
Appendign Program Files from Disk
Correction to Commodore Floppy Disk User Manual (2040 & 3040)
DOS Rename Failure

PET IN EDUCATION
- Educational Software
- TVA Meter for the Physics Lab

Sorting by Insertion and Chaining
Re-Dimensioning Arrays
PET in Medicine
Its all happening at the Poly
Beginning Machine Code

We would like to thank John Kyle Price and the Bristol Software Factory for allowing us to reproduce their artwork on the cover of CPUCN.

Commodore News

Andrew Goltz - Editor CPUCN

As preparations for the PET Show move into top gear all those concerned agree that this will be the most important event in the PET world since the launch of the PET over 2 years ago.

Packed into London's Cafe Royal will be the largest collection of PET hardware and software products ever gathered together under one roof. There will be ample opportunity to see demonstrated Commodore's own programs from both the Business Software range and the Petpack Master Library. In addition nearly 30 independent companies will be exhibiting hardware and software products and some of these will be launching new products at the Show.

Commodore will be unveiling for the very first time in the UK the new 8032 series large screen PETs and the complementary 8050 1-megabyte disk drives. These two new products are intended to secure Commodore's position at the top end of the microcomputer market, and are expected to be released in the UK later this year.

A comprehensive program of application seminars will provide an opportunity to explore many of the key areas in which PETs are in every day use. For the more technically minded, there will be an opportunity in the special Users Club seminars to quiz Commodore's resident boffins.

A special feature of the Show is the space allocated to local User Groups. Facilities will be available for PET Users to become acquainted with other PET Users in their area and to encourage the formation of new user groups.

All secretaries of PET User Groups which intend to be represented at the Show should write to me as soon as possible. I should also be glad to hear from anyone interested in helping to form local user groups in areas where no such groups exist at the moment

Don't forget to mark the Pet Show dates 13th and 14th June in your diary. See you there!

All Commodore Pet User Club Members will be invited to attend the Pet Show, free of charge, as guests of Commodore. A free show-guide and complimentary ticket will be distributed with CPUCN 2.6.

- - -

A new ambitious programme of courses and seminars will be run by Commodre's Training Department during the second part of the year. New Subjects include: "PET in Accounting", "Word Processing" and "Service and Maintenance". Full details in the latest training brochure, enclosed with this newsletter.

Teach-in

Mike Gross-Niklaus
Commodore Training Manager

One purpose of "TEACH IN" is to bring you the latest news on the 'training front'. The column will also carry the kind of article I was writing in "PRINTOUT" together with any contributions you care to make. You can write to me at:-

Commodore Systems Division
Attention of the Training Manager
360 Euston Road
LONDON NW1

Our latest training brochure should be enclosed with this copy of CPUCN, further copies are available from our Euston Road office. It's difficult in a brochure to give you the complete picture, particularly on what it's like to come on one of the courses. Hopefully these notes will convey some of the atmosphere of Commodore Training Courses.

WHY HOTELS?

They are run in hotels to give all participating the 'luxury' of being able to concentrate on one subject without distraction and to enable the use of the evening hours to get as much done as possible within the two or three day span of the course.

THE DAILY TIME-TABLE.

The typical course day starts at 9 am, breaks for lunch at 1pm, starts again at 2pm and finishes for the afternoon at 5pm. Dinner is at 7pm and we start up again at 8 pm and go on to 9.30pm. That's the theory, arrived at after suggestions from students in many course appraisal discussions. In practice, it doesn't work out quite like that! On bringing the afternoon to a close at 5 pm, instead of the clatter of students rushing for the nearest bar, David Pocock and I are deafened by the clatter of keyboards as students, apparently glued to their chairs, continue with their own extensions to the course exercises. The same applies in the evenings after 9.30pm.

USE OF GAMES PROGRAMS.

All the courses use disk drives to speed up the process of program loading. Around 9.30pm David produces the now famous Training Department games disks which include a fair selection of the published games and quite a few that are not yet on general release. One of the best ways of improving your BASIC is to list and read through such games, because a very wide range of useful techniques are used, and being in games, their purpose is readily understood.

We are frequently surprised by the number of students, who, while relaxing after the day's work playing through some of the games, very soon are busy writing and shortly after that keying in their own programs, using techniques and ideas from the games programs.

It can certainly pay dividends! On a recent PRIMARY course, two students without any previous knowledge of PET or BASIC, wrote a beautifully laid out version of MASTER MIND with excellent screen formatting without any major advice from us tutors. On the same course an accountant, with immense enjoyment and satisfaction, took the elementary game of "GUESS A NUMBER" and turned it into one of the most humorous programs I've seen.

If you frown a bit at the emphasis on games, remember that the requirements of a game are easier to formulate and understand than the average business program and yet the techniques used in coding them are applicable to all types of program. One good example of this is the use of Guess A Number techniques in setting up a binary search for arrays. Another is the use of games graphics to neaten up the presentation of a program, so important if lay staff are to use it correctly day after day.

PACING THE COURSE.

Obviously some people work faster than others. We deal with the difference in pace in two ways. During the practical sessions which occupy over 50% of the course time, David and I spend more time with those who are temporarily 'hung up'. For the real 'fighter pilot aces', there are additional exercises which extend beyond the normal limits of the course.

WHAT DO YOU GO AWAY WITH?

Everybody achieves something on the courses. I believe it's fair to say that of the several hundred people who have attended so far, not one has wasted their time. On the "Basic for Beginners" course it's very occasionally obvious to us and to a particular student that programming isn't their real forte, but even in these cases, the student takes away small programs that he or she has written, and can take pride in.

The majority of students successfully complete the major exercise at the end of each course. On the Basic for Beginners course it's a program which allows the updating of a monthly orders book for one product, and neat display of the totals on the screen. For the 'Fighter Aces', this is extended to deal with several products.

On the Disk Utilisation course, the final exercise is to write a program which creates a direct access data base with three unlabelled items per record. The extension of this for the high flyers is a data base with a specified number of items per record, each with a label.

On the Advanced Programming with Basic

course, students write modules of a model job throughout the course, ending up with a professionally laid out, annotated program complete with specification and flowcharts. The program allows the creation of a record array with three labelled items per record, which can be sorted on any item, stored on tape or disk, recovered, of course, and printed in sorted sequence. The extension of this exercise is to write additional modules, such as selective printing of records or use of the Shell-Metzner and other fast sorts.

And, naturally, students for all courses take away with them the manuals and reference sheets. These include the exercise specifications and typical solutions, including that for the main exercise. Past students tell us that they use these as reference information more than any other document or published book.

WHAT ABOUT THE SCIENTISTS?

Perhaps surprisingly, we find that those looking towards mathematical programming need the same techniques as those used for commercial programs. Most engineers and scientists who have attended our courses have a pretty good idea of the algorithms they want to use and it takes relatively little time for them to learn how to achieve this using Basic. Their main problems are the usual ones of effective input and output, optimum use of disk, proper specification and layout of the programs.

WILL YOU BE AN EXPERT AFTER THE COURSE?

No! Basic is a language. The courses show you the 'vocabulary', the 'grammar' and some of the idiomatic 'phrases'. As with any other language the only way to become fluent is to converse in it. The Commodore courses set you up to do this. At the end of each course I advise students to 'keep it simple' to start with: that is to say, to specify and write simple programs which achieve no more than could probably be done, without much thought, on paper. The reason for this is to maintain the enthusiasm and confidence not only of the student, but also that of his collegues who want to see what's going to happen now he's back from the course.

Nothing kills your confidence and the support of your work colleagues like a large scale computing project which flops! Far better to write a little routine which allows you to feed in the latest dollar conversion rate each day and prints out a conversion table for each department in the firm who needs one. Incidentally, that's one of the exercises on the "Basic for Beginners" course!

Microcomputers and the Businessman

Question: what have the following in common?

- A prehistoric man;
- An unhappy employee with a pocket magnet;
- A machine which claims to be READY, when it isn't;
- A potential user of a computer package who spends an hour trying to do all the things wrong in operating it which could conceivably be done wrong in a year of normal operation;
- Petrol at 3p per gallon;

Answer: They are all discussed at the Commodore Businessmen's seminars.

What are the Commodore Businessman's seminars really like?

This question is likely to arise more frequently as these appreciation seminars continue.

The answer is not so simple as it might seem.

To start with, they might perhaps be called "Microcomputers and the Business Person", since there is usually at least one lady present!

The background of the participants is very diverse, and as a result, their needs vary greatly: Some are from businesses with mainframe machines, looking to see how the microcomputer can supplement existing facilities. Sometimes they are owners of Commodore machines, looking for ways to obtain full benefit from their equipment. Other delegates are considering the use of microcomputers in their business, and want to know how to go about the difficult task of selection of both hardware and software.

The seminars are constructed somewhat flexibly so as to allow for these various aims and backgrounds. In particular the question-and-answer sessions are geared to the needs of the particular group which is present on any one occasion.

The day starts with a general introduction to computers, with a little necessary history being followed by a discussion of the characteristics and capabilities of the hardware and software.

Jargon is explained, and the mystery surrounding the subject dispelled.

Practical demonstrations follow this first talk so as to give preliminary exposure to the hardware.

The second session leads into the software area, with information as to what such machines will accommodate, and what they will not. The advantages and disadvantages of off-the-shelf packages, modified packages, and bespoke programs are given full coverage.

Further demonstrations of packages follow, some being re-tailored to suit applications suggested by participants.

The third session deals with the ways of deciding which applications are suitable for computerisation, and, more importantly, which are not. Assessing both the suppliers of hardware and software, and their products themselves is covered. Problems of privacy and security are also given due consideration.

Demonstrations of programs which users might write for themselves are given, and the problems likely to be encountered are discussed.

The final session consists of a question-and-answer period, when any question is given the full treatment by all the speakers. At this stage, the diverse backgrounds of the participants shows the most; questions range from whether it is possible to hook-up two PETs overland to communicate with each other, to "How can I get my dealer to show me the xxx program actually working"!

The emphasis is that the microcomputer has a part to play in many modern businesses, that, as with all tools, there is a right way and a wrong way to use it, and that there are pitfalls for thg unwary and the inexperienced.

Perhaps a surprising feature is that the day is far from being a festival of total enthusiasm for the micro; still less is it entirely Commodore-orientated. The use of outside speakers ensures a degree of impartiality, and although their frankness may occasionally make the Commodore representatives stir uneasily in their seats, it is in keeping with the deliberate policy of Commodore's Training Manager, Mike Gross-Niklaus to provide a good service to his customers and not just mount another "sales exercise".

It is felt that the visitor to the seminar will be much better able to make an informed choice of microcomputer equipment, and of suitable applications for that equipment after attending the seminar.

It would be wrong to assume that these seminars are deadly serious throughout. Although the subject-matter is serious, The speakers go out of their way to keep the presentation light and, so far as possible, humourous. The examples given at the beginning of this article give some idea of the variety of images used to put across the material. Frequently cartoons are projected on the screen to emphasize a point, and numerous humourous anecdotes are recalled: some may even be true stories, taken from life!

The feeling which comes across is that a group of friends is meeting to discuss matters of common interest and importance, in comfortable conditions, and a relaxed and friendly atmosphere.

Next time your business neighbour drops in for a friendly chat about your PET computer - don't just show off 3-D Startrek, give the chap a chance - show him this article as well!

Appending program files from disk

It is very useful to be able to append portions of program from a subroutine library and even more useful if these can be readily pulled in from disk. Unfortunately, the Append command in Commodore's Disk Utility Maintenance package will only work with data files. In order to append program files it is necessary to get rid of the zeroes which precede the End of File marker and indicate the end of one program and also the starting address which indicates the beginning of the other.

The routine listed below will carry out these functions, and has been successfully used to append the test routine to Mr. Slow's sort program which appears elsewhere.

```
10 INPUT"1ST SOURCE FILE NAME";A$
20 INPUT"2ND SOURCE FILE NAME";B$
30 OPEN15,8,15
40 PRINT#15,"I0"
50 PRINT#15,"I1"
60 OPEN5,8,5,A$+",P,R"
65 GOSUB1000
66 INPUT"DESTINATION DRIVE#";S$
70 INPUT"DESTINATION FILE NAME";C$
80 OPEN6,8,6,S$+":"+C$+",P,W"
90 GOSUB1000
100 GET#5,D$:IFST<>0THEN160
105 IFD$=""THEND$=CHR$(0)
110 IF F THENPRINT#6,E$;
120 F=-1:E$=D$:GOTO100
160 CLOSE5
180 OPEN5,8,5,B$+",P,R"
190 GOSUB1000
200 GET#5,A$:GET#5,A$
210 GET#5,D$:T=ST
215 IFD$=""THEND$=CHR$(0)
220 PRINT#6,D$;
230 IFT=0THEN210
240 CLOSE5:CLOSE6:CLOSE15
250 END
1000 INPUT#15,ER,ER$,ET,ES
1010 IFER<>0THEN1050
1020 RETURN
1050 PRINTER,ER$,ET,ES
1060 STOP
READY.
```

Correction to Commodore Floppy Disk User Manual (2040 & 3040)

Page 17 & 29
SAVE and OPEN with REPLACE

Reference: SAVE"@dr:fn:",dn
OPEN3,8,5,"@0:JDATA,USR,WRITE

Do not use SAVE with Replace or OPEN with Replace (e.g. SAVE"@0:"FILENAME",8). These commands have a bug which can cause other files to be corrupted on the disk.

Options:

1. Scratch old file first then save new file.
2. - Save new file under a dummy name.
   - Scratch old file.
   - Rename dummy file to correct name.
3. Save file on other drive and use old file as a back-up.

Page 16 and 17
SAVE and OPEN Write File without Drive No

The DOS searches for the filename starting on the most recently used drive. If it is not found on either drive as an existing file or as an unclosed file,then the DOS assigns the file entry to the most recently used drive. The contents, however, are placed on the opposite drive.

The result is a file entry which is linked to space on the diskette which may be used by other files or not at all. The BAM of the opposite drive is updated by the DOS.

The contents of the other files on each diskette are not altered, but care should be taken in the recovery procedure. If either drive is empty or un-INITIALIZED, the process will halt, leaving the file unclosed.

Recovery Procedure:

The safest way to recover is to properly save the file on another diskette, and copy all files of interest from the diskette with the bad entry to the other diskette. The corrupted diskette may be restored with the verify command.

An easier but less guaranteed method is to scratch the file entry and then restore both disks with the verify command.

Page 19 (and 25)
Formatting With Write Protect Tab On (NEW with ID or Duplicate)

If an attampt is made to format a disk which has a Write Protect Tab in place, the system will attempt to write. After the system detects that no writing is taking place, an error condition is generated, but the write signal line is left on. Any subsequent operation will result in writing to the drive involved in the operation.

Since the write protect is also a hardware disable, any protected diskette will not be disturbed. To recover from this situation, power reset the disk unit.

Page 25
DUPLICATE-Write Error

If a write error is encountered in the DUP command, the disk unit will attempt the write continuously. This indicates probable media failure and the diskette should be discarded. This problem may be detected by watching the R/W head or listening carefully to the disk. If the click sound of the head changing tracks is not noticed after 1 minute, then more than likely the problem has occurred.

Page 25
VALIDATE

The manual states that the diskette should be INITIALIZED immediately after a Validate error. If this is not done, a loss of file contents will be inevitable, since the BAM in memory is left in an indeterminate state.

Page 29
RENAME-File Not Renamed

This problem has been observed on occasions. Usually adding a file to the diskette will allow RENAME to work. Also, copying the file to the new name will also work.

Page 43
BLOCK-ALLOCATE and BLOCK-FREE, Illegal Track or Sector Requested

The B-A & B-F commands will generate an error message which overlays part of the previous message. The position is dependent on the length of the incoming command. For example, if the previous message was 00, OK, 00, 00 and the command was 38 characters long, subsequent inputs would still access the previous message.

Care should be taken when transmitting these two commands. It should be as short as possible and Track & Sector should be legal values. The following table indicates the legal ranges.

Track	Sector
1-17	0-20
18-24	0-19
25-30	0-17
31-35	0-16

Page 43
BUFFER-POINTER at 0

The B commands are intended for Record oriented disk access. Since position 0 is used as the pointer to the last valid data byte in the record, it is not normally accessed. If it is necessary to access this byte, (for gaining access to a track link for example) B-P at 0 will allow access (GET# or PRINT#) only if the last character pointer is not 255.

Solution:

```
OPEN1,8,15
OPEN2,8,2,"#0"
PRINT#1,"M-R"CHR$(0)CHR$(17)
GET#1,A$:REM 1st Byte of Buffer
```

Page 44
MEMORY-READ GET # without EOI

The Memory-Read command is intended to provide an unlimited access to any part of the file interface controller's memory. The byte read from the memory is placed in the error buffer and the character count is set to one. EOI is not sent with the byte. Consequently, an INPUT# from the Error channel (SA=15) will not terminate. If M-R is to be executed, only a GET# should be used in accessing the byte.

DOS Rename failure

Harry Broomhall has contributed the following notes about the DOS RENAME failure.

If the following conditions are true, RENAME will not work for any file in the directory :-

1. The last sector in the chain of directory sectors contains either no directory entries or only scratched entries.

2. In any previous sector of the directory there is at least one scratched entry.

Harry has found when both conditions are fulfilled, RENAME will appear to work, giving "OK" as the response down the error channel but, in fact, the RENAME process has not taken place. The only easy cure is to copy the old file providing it with a new name and then scratching the original file.

To recover from this condition, copy all files onto a new disk using the Commodore utility, Copy Disk Files. This will work provided all files are of the non-User type.

Direct Access

In CPUCN 2.4 Mike Gross-Niklaus discussed the components of a direct access program. In this issue, we print a listing of an example program written by Mike. As well as demonstrating Direct Access, the program also illustrates the "blocking" convention and principles of clear annotation which are taught on Mike's programming courses. The only changes made to the program are the addition of asterisk REM statements to make the photo-reduced listing easier to follow.

```
10 REM UNI
20 REM MIKE GROSS-NIKLAUS
30 REM 6/11/79
40 REM DISK TECHNIQUES PRACTICE ONLY.
995 REM   *   *   *   *   *
1000 REM PREL
1010 DIMA$(20,2)
1020 R$=CHR$(13)
1040 P=0:REM INDEX POINTER
1050 OPEN15,8,15:REM CMD CHAN
1060 OPEN2,8,2,"#":REM D/A FILE
1095 REM   *   *   *   *   *
1100 REM RID ARRAY OF NULL STRINGS
1110 FORI=0TO20:FORJ=0TO2:A$(I,J)=CHR$(160):NEXTJ,I
1995 REM   *   *   *   *   *
2000 REM MENU
2010 PRINT"■■MENU OF COMMANDS"
2020 PRINT"■A = ADD     RECORD TO   DISK 1
2030 PRINT"■D = DISPLAY RECORD FROM DISK 1
2040 PRINT"■W = WRITE   INDEX  TO   DISK 0
2050 PRINT"■R = READ    INDEX  FROM DISK 0
2060 PRINT"■L = LIST    INDEX  FROM ARRAY
2070 PRINT"■Q = QUIT
2080 PRINT"■■PRESS A, D, W, L OR Q
2090 GETA$:IFA$=""THEN2090
2095 REM   *   *   *   *   *
2100 REM BRANCH ON COMMAND
2110 Z$="ADWRLQ"
2120 FORI=1TOLEN(Z$):C$=MID$(Z$,I,1)
2130 IFA$=C$THEN2150
2140 NEXT:GOTO2000
2150 ONIGOSUB3000,4000,5000,6000,7000,9000
2160 GOTO2000
2995 REM   *   *   *   *   *
3000 REM ADD A RECORD
3010 GOSUB42000:REM GET KEY WORD
3020 PRINT"■■DETAILS":PRINT:INPUTD$
3095 REM   *   *   *   *   *
3100 REM FIND NEXT FREE BLOCK ON DISK 1
```

```
3110 T=1:S=0
3120 PRINT#15,"B-A";1;T;S:GOSUB41000
3130 IFEN=0THEN3200
3140 T=ET:S=ES:GOTO3120
3195 REM   *   *   *   *   *
3200 REM WRITE K$,D$ TO DISK 1
3210 PRINT#15,"B-P";2;1
3220 PRINT#2,K$;R$;D$;R$
3230 PRINT#15,"B-W";2;1;T;S
3295 REM   *   *   *   *   *
3300 REM UPDATE INDEX
3310 A$(P,0)=K$:A$(P,1)=STR$(T):A$(P,2)=STR$(S)
3320 P=P+1:IFP<21THENRETURN
3395 REM   *   *   *   *   *
3400 REM INDEX FULL
3410 PRINT"■■INDEX FULL. WRITING TO DISK"
3420 GOSUB5000
3430 GOTO9000
3995 REM   *   *   *   *   *
4000 REM SEARCH INDEX FOR A RECORD
4010 GOSUB42000
4020 FORI=0TOP-1:IFA$(I,0)=K$THEN4100
4030 NEXT
4040 PRINT"■■KEY-WORD NOT IN INDEX"
4050 FORT=1TO2500:NEXT:RETURN
4095 REM   *   *   *   *   *
4100 REM GET RECORD & DISPLAY
4110 T=VAL(A$(I,1)):S=VAL(A$(I,2))
4120 PRINT#15,"U1";2;1;T;S
4130 PRINT#15,"B-P";2;1
4140 INPUT#2,K$,D$
4150 PRINT"■■KEY-WORD■ "K$
4160 PRINT"■■DETAILS":PRINT:PRINTD$
4170 PRINT"■■■PRESS SPACE TO REGAIN MENU"
4180 GETA$:IFA$<>" "THEN4180
4190 RETURN
4995 REM   *   *   *   *   *
5000 REM WRITE AN INDEX
5010 INPUT"■■NAME FOR INDEX■";F$
5020 DR$="0:":PRINT"■■OVERWRITE?"
5030 GETA$:IFA$<>"Y"ANDA$<>"N"THEN5030
5040 IFA$="Y"THENDR$="@0:"
5050 OPEN3,8,3,DR$+F$+",S,W":GOSUB41000
5060 PRINT#3,P;R$;
5070 FORI=0TOP-1:FORJ=0TO2:PRINT#3,A$(I,J);R$;:NEXTJ,I
5080 CLOSE3:RETURN
5995 REM   *   *   *   *   *
6000 REM READ AN INDEX
6010 INPUT"■■NAME FOR INDEX■";F$
6020 OPEN3,8,3,"0:"+F$+",S,R"
6030 INPUT#3,P
6040 FORI=0TOP-1:FORJ=0TO2:INPUT#3,A$(I,J):NEXTJ,I
6050 CLOSE3:RETURN
6995 REM   *   *   *   *   *
7000 REM LIST INDEX
7010 PRINT"■INDEX LISTING FOR "F$:PRINT
7020 FORI=0TOP-1:PRINTI,:FORJ=0TO2:PRINTA$(I,J),:NEXTJ,I
7030 PRINT"■■PRESS SPACE TO REGAIN MENU";
7040 GETA$:IFA$<>" "THEN7040
7050 RETURN
8995 REM   *   *   *   *   *
9000 REM CLOSEDOWN
9010 CLOSE2:CLOSE3:CLOSE15
9020 END
40995 REM   *   *   *   *   *
41000 REM READ CHAN 15 MESSAGES
41010 INPUT#15,EN,EM$,ET,ES
41020 IFEN=0OREN=65THENRETURN
41030 PRINT"■■■DISK ERROR":PRINT
41040 PRINTEN;EM$;ET;ES
41050 CLOSE3:CLOSE2:CLOSE15:END
41995 REM   *   *   *   *   *
42000 REM INPUT KEY WORD
42010 INPUT"■■KEY WORD■";K$
42020 RETURN
READY.
```

Educational Software

Pete Gerrard - Cassette Library Manager

As you may be aware, the original 8K Pet with an integral cassette deck was at one time taken off the market. However, the demand for the return of this particular model, especially from schools and colleges was so great - it is compact, robust and reasonably portable - that has been re-introduced at the new low price of £475. Obviously of concern to educationalists is the amount of good quality educational software available - a computer is only as good as its programs. With this in mind Commodore have released a number of educational packages. In the next issue of the CPUCN I shall be reviewing some of our NEW releases in this area, but in this issue I want to concentrate on the products already available from Commodore.

PROGRAMS FOR COMPUTER STUDIES

Programs designed to initiate the first-time user into the use of the Pet, such as BASIC BASIC (MP001), SQUIGGLE & BIG TIME (MP002), SNARK (MP026) and STRATHCLYDE BASIC COURSE (MP039), do not appear in our Master Library catalogue in the section devoted to educational software, however, if a school or college is intending to run a course in computer studies they are all extremely useful products. Even if the PET computer is mainly used for Computer Assisted Learning of subjects other than computing, it is useful to have such programs to hand in case a programming query crops up, particularly the excellent STRATHCLYDE BASIC (MP039).

As with all our products our policy with educational programs is to provide good value for money. Usually four or more programs are provided on a £10 pack, so that the most that has to be paid for an individual program is £2.50. This 'rule' is very occasionally broken in the case of an exceptionally good program. We are aware of the financial problems experienced by educational establishments!

PHYSICS PACKS

The first of our educational packages to be introduced was PHYSICS PACK 1 (MP046). This is a series of Physics tutorial programs, which provide simulations of some standard experiments. In the first program, a pair of Helmholtz coils are graphically reproduced bending an electron beam, an experiment which enables e/m to be calculated for the electron. Then, we have a program which simulates a decay process using random numbers, plotting both linear and non-linear graphs. The third program treats Schrodinger's method as a simulation where the student has to find the correct value of E (the electron energy.). The final program consists of two tutorials on the Conservation of Momentum, and allows the student to type in his or her own values of mass and velocity.

Since the release of MP046, a number of other programs have entered the Petpack Master Library. One of these was PHYSICS PACK 2, (MP053), which contains four programs. The first, Electron, covers the important topic of the electronic structure of atoms and chemical bonding, presenting a graphical illustration of these topics. This is followed by a series of questions on the subject matter covered. The third program is concerned with the properties and formulae of a group of compounds called Alkanes - an important subject these days, the cost of petrol being what it is! The final program deals with the classic Millikan's experiment, providing a fine simulation of the apparatus used in the experiment. The experiment is used to determine the charge on the electron, and the use of this program considerably speeds up the time usually spent on the experiment, as well as saving the purchase cost of the apparatus!

LANGUAGES

Before moving onto our mathematical aids, I would like to mention a program called LANGUAGRES, (MP064), which provides SIX programs for £10. These cover the following foreign languages :- Danish, Dutch, Spanish, French, Italian and German. The EEC basically! Each program provides up to one hundred words OR PHRASES in the foreign language of your choice, and you are asked to give the English translation of the word(s) presented. Alternatively you can be given the English and asked for its foreign equivalent, or finally a random mixture of both. Where necessary you will be asked to give the correct gender also. After you have finished the exercise a score and comment will be made on your performance, and when a suitable standard has been attained you are requested to modify the program! This enables you to insert more difficult words, phrases, verbs etc. and thus gradually raise the level of your, or indeed your pupil's, proficiency. Consequently the programs can be started at a beginner's level and gradually modified right up to the requirements of a highly advanced pupil, giving a thorough grounding in any one of the six languages.

MATHS PACKS

At present there are three maths packages designed to be used as teaching aids, but as with the Physics Packs, and indeed educational material generally, this is an area in which we are actively engaged in expansion.

INTRODUCTION TO ALGEBRA (MP061) consists of thirty one programs in a £20 package. The programs cover ALL algebraic material encountered up to O-Level standard, and are extremely useful for tuition and revision of topics in algebra. The programs start at an elementary level, and gradually work up through a series of tutorials and question and answer sessions to encompass

everything the student needs to know before sitting the actual O-Level examination. All the programs are very well written, and the information is presented in a manner designed to put any pupil at their ease as they work their way through the programs. Altogether a very competent package.

TEACHING NUMERIC SKILL TO YOUNGER CHILDREN

MATHEMATICAL GAMES (MP062) is not pitched at any particular topic in mathematics, but instead aims to teach numerical skill generally in a friendly and informal manner. One successful method of teaching is to treat the subject matter as a game. Younger children in particular seem to respond well to this approach, and if at the same time they come to regard the computer not as an impersonal machine but rather as a friendly tool, so much the better. The topics that this particular set of eight programs (the package costs £10) cover, include place value, multiplication, addition, subtraction and division, and the general grasping of what numbers are all about. Learning can be fun!

STATISTICS

The final program in our mathematical series is called SAMPLING, and costs £10. Although only one program, it consists of six different 'units' within that program. As might be gathered from the title, the program is concerned with sampling and sample distributions, as it simulates the drawing of a random sample from a discrete probability distribution. The sample values are tabulated, and/or displayed in the form of a histogram, and the sample mean and sample standard distribution are calculated and displayed, along with the population parameters for comparison. As it caters for six different distributions, the program can be used to illustrate many aspects of a course on probability and statistics. Visual displays are far easier to grasp than any amount of text and equations! The distributions catered for are:- uniform, user-definable, uniform on the digits nought to nine, Binomial, Poisson and Geometric.

That finishes our current list of educational titles, which will be considerably expanded in the near future. The next newsletter will contain descriptions of some of those future releases. In the meantime however there are a number of other programs in our Petpack library that, whilst not being written with the educational market in mind, are eminently suitable for use in that area.

OTHER MASTER LIBRARY TITLES

Such programs as BASIC MATHS PACKAGE (MP007) £15, BASIC STATISTICS PACKAGES one and two (MP008 & MP030) £15 each, and BEST LINE/LEAST SQUARES (MP055) - £10) could all be used by teachers.

The science and engineering part of our library also contains some very useful programs. For instance, LINEAR CIRCUIT ANALYSIS (MP047) £10, provides the following functions on two programs called WALLAP and AL CAPONE. Al Capone analyses general linear circuits at many frequencies over a specified range, and the circuits may incorporate inductors, capacitors, resistors, transistors and amplifiers. In other words a quite general linear circuit analysis program. Wallap is also a circuit analysis program, but instead of giving numerical results it expresses circuit response algebraically. Although restricted to passive circuits, Wallap is in all other senses quite general. An extremely useful program.

FAST FOURIER TRANSFORMS (MP058) £15, is a set of six programs adapted to run on the Pet and employing well-established routines for the FFT, and is intended to serve as a tutorial introduction to the use of the FFT algorithm. Since FFT covers the analysis of speech and music, electronic signal processing, digital filter design, synthetic aperture radar, time series analysis, analysis of seismic waves, vibration analysis, infra-red spectroscopy, radio interferometry AND SO ON, one can quickly appreciate the uses of such a package at sixth form level in schools and also in colleges!

Together with such programs as LOGICA (enabling you to set up a circuit in memory and work out the outputs for any set of inputs, and modifying, testing and storing ciruits) and KARNAUGH MAPPING (which allows you to set up and modify Boolean circuits of up to six variables by the graphical method of Karnaugh mapping), which come on MP056 at £10; MECHANICS OF MATERIALS (a set of twelve programs covering the analysis of materials, with a detailed handbook), etc., it can be seen that Commodore Petpack Master Library has a lot to offer for the educational market.

A description of ALL our programs can be found in Petpack Master Library Catalogue. Copies can be obtained through your local Comodore Dealer or from the Commodore Information Centre, 360 Euston Road, London, NW1 3BL.

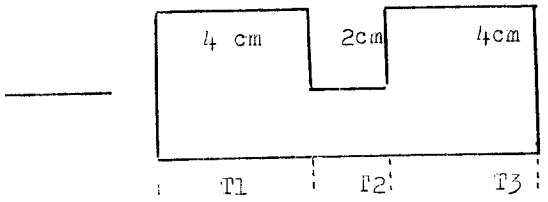
TVA Meter for the Physics Lab

By Bob Sparkes

This application enables the user to measure time intervals up to 6.5 seconds in units of 100 μs and to display the result on the PET screen in large letters. The measurement can be displayed with 2, 3, or 4 significant figures and a decimal point and up to four measurements can be stored internally and recalled later. This enables the user to replace the "Millisecond Timers" at present required for this purpose, and, since these cost some £200, represents a considerable saving; only a few pounds worth of additional equipment is needed.

For example, this equipment could measure the time required for a 10-cm card on a trolley to pass in front of a photo-cell. From this the velocity of the card may be calculated in the normal way. But why not let PET do the calculation automatically? The resulting velocity can then also be displayed to the whole class and we have the "Velocity Meter" for which Physics teachers have waited so long.

Furthermore, by using a "double" card, PET can measure the three time intervals T1, T2 and T3 and compute the acceleration from these figures.

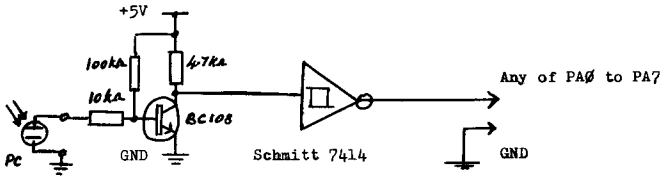


The acceleration is given by:
A = 0.04 x (1/T3 - 1/T1)/(T2 + T1/2 + T3/2), which is derived from the familiar a = (v - u)/t.

The program is in BASIC for the calculations and in machine code for the measurements and display. The display itself is worthy of consideration, since it could easily be adapted to a display of any alpha-numeric character in large letters, and could therefore be used to communicate with the whole class at once. A discussion of this display program is given separately.

The measurement is simple. The photocell is connected to the User Port via a simple interfacing circuit and PET reads the input (decimal location 59471) and compares this with its previously recorded state. Any change in any of the inputs causes a branch to the timing mode, which continues until another change produces a stop. For acceleration three consecutive timing loops are required, so the number of loops is POKED into a convenient location prior to the SYS command. With this system the timer can be started in one place by one photocell and stopped in a different place by another. Personally I found two photocells to be sufficient, but up to eight could be used. The inputs are also able to detect increases or decreases in the logic level, so the facility on most commercial timers of "Make to Start - Break to Stop" etc. is automatically built in.

USER PORT INTERFACE DETAILS



PROGRAM DETAILS

For those without machine code experience, the following BASIC program will load the machine code program also. This program was not planned before being written (it just grew from one idea to the next) so the expert programmer will find many instances where space may be saved.

BASIC listing.

```
 1Ø   GOSUB3ØØØ
15    W=1ØØ:U=1Ø1:L=119:H=116:K=Ø
2Ø    PRINT"clr TIMER, VELOCITY AND ACCELERATION METER"
4Ø    PRINT"cdown FOR ACCELERATION, PRESS A"
5Ø    PRINT"cdown FOR VELOCITY, PRESS V"
6Ø    PRINT"cdown FOR TIME INTERVAL, PRESS T"
65    GETAØ:IF AØ=""THEN 65
7Ø    IF AØ="A"THEN1ØØ
8Ø    IF AØ="V"THEN3ØØ
9Ø    IF AØ<>"T"THEN65
1ØØ   GOTO 35Ø
1Ø5   PRINT"clr MEASURING ACCELERATION"
1Ø6   GOSUB 1ØØØ
```

```
109       S=2:POKE251,3:SYS(920)
110   T1=PEEK(1001)*256 + PEEK(1000)
120   T3=PEEK(1005)*256 + PEEK(1004)
130   T2=PEEK(1003)*256 + PEEK(1002)
140   IF T1=0     OR T3=0 OR T2=0 THEN 200
150   T2=T2 +(T1+T3)/2
160   Q=4000000*(1/T3 - 1/T1)/T2

170    A(K)=Q
180    GOTO 500
200    REM  ERROR ROUTINE
210    PRINT"cdown I DONT INTEND TO WAIT ALL DAY"
220    PRINT"cdown PRESS T TO TRY AGAIN"
223    PRINT"cdown PRESS M TO RECALL PREVIOUS READINGS"
225    K=0
230    GET C$:IF C$<>"M" AND C$<>"T" THEN 230
235    IF C$="M" AND A$="V" THEN800
236    IFCC$="M" AND A$="A" THEN900
237    IF C$="M" AND A$="T" THEN950
240                    GOTO   15

300    PRINT"clr MEASURING VELOCITY"
305    GOSUB1000
309        S=1:POKE  251,1:SYS(920)
310    T1=PEEK(1001)*256 + PEEK(1000)
320    IF T1 = 0 THEN 200
330    Q =1000/T1:V(K)=Q
340    GOTO 500

350    PRINT"clr MEASURING TIME INTERVAL"
355    GOSUB 1000
359        S=1:POKE 251,1:SYS(920)
360    T1=PEEK(1001)*256 + PEEK(1000)
370    IF T1=0 THEN 200
380    Q=T1*0.0001: T(K)=Q
390    GOTO 500

500    REM  DIGIT SEPARATION
505    PRINT"clr":DP=0:SN=0
510    IF Q<0THEN Q=Q*-1:SN=1
520    IF Q>=1THENQ=Q/10:DP=DP+1:GOTO520
540    FOR I=0TO3:D=INT(Q*10)
550    Q=Q*10 - D
560    IF I<DP THEN D(I)=D
570    IF I>=DP THEN D(I+1)=D
580    NEXT
590    D(DP)=0
600    IF SN=1 THEN GOSUB 700
603    PRINT"clr":FORN=0TO4:POKE251,N:POKE252,D(N):SYS(662):NEXT
605    IFA$="T" THEN SYS(648):GOTO660
610     SYS(634):SYS(648)
```

```
676   0
677   165                 LDA DIGITVAL          Get digit to be displayed
678   252
679   10                  ASL                   Multiply by 8
680   10                  ASL
681   10                  ASL
682   168                 TAY                   Pointer to table of BITS
683   185   BITSLOOP      LDA BITSTABLE,Y       Get BITS
684   32
685   3
686   157                 STA TEMP,X            Keep BITS temporarily
687   232
688   3
689   200                 INY                   UPdate to next row of bits
690   232                 INX                   UPdate to next TEMP store
691   224                 CPX #  8              8 rows done ?
692   8
693   208                 BNE BITSLOOP
694   244
695   160                 LDY #  -32            Yes, send bits to screen
696   223
697   162                 LDX #  -1
698   255
699   232   NEXTROW       INX
700   224                 CPX #  8              8 rows done ?
701   8
702   208                 BNE  BITSEND
703   1
704   96                  RTS                   Yes; finished.
705   169   BITSEND       LDA #        8
706   8
707   133                 STA BITCOUNT
708   251
709   24                  CLC
710   152                 TYA                   Move pointer to start of next
711   105                 ADC #  32             row
712   32
713   168                 TAY
714   169                 LDA #  160            "White square" code
715   160
716   200   NEXTBIT       INY
717   30                  ASL TEMP,X            Test MS Bit for logic 1
718   232
719   3
720   144                 BCC  NOBIT
721   2
722   145                 STA (SCNPOS),Y        Send "white square" to screen
723   253
724   198   NOBIT         DEC  BITCOUNT
725   251
726   240                 BEQ  NEXTROW          8 bits done, go to next row
727   227
```

```
62Ø    POKE 33514,L:POKE 33515,L
63Ø    IFS=1 THEN POKE 33477,H:POKE 33517,H
64Ø    IFS=2 THENPOKE 33437,W:POKE33477,W:POKE33478,U:POKE  33517,U:POKE33557,99
66Ø    K=K+1: IFK=F THEN 22Ø
665    PRINT"home READY FOR NEXT READING"
67Ø IFA$="A" THEN 109
675    IFA$="T"THEN 359
68Ø    GOTO 3Ø9
7ØØ    FOR I=4TO1STEP-1:D(I)=D(I-1):NEXT
71Ø    D(Ø)=11
72Ø    RETURN
8ØØ    PRINT"clr": FOR K=Ø TO F-1:PRINT" V"(K+1)"    =    "VAL(LEFT$(STR$(V(K)),6)):
       NEXT
81Ø    PRINT"cdown PRESS R TO RESTART"
82Ø    GETD$:IFD$<>"R" THEN82Ø
83Ø    GOTO 15
9ØØ    PRINT "clr":FOR K=ØTO F-1:PRINT"A"(K+1)"  =  "VAL(LEFT$(STR$(A(K)),6)):NEXT
91Ø    GOTO 81Ø
95Ø    PRINT"clr":FORK=ØTO F-1:PRINT"T"(K+1)"  =  "VAL(LEFT$(STR$(T(K)),6)):NEXT
96Ø    GOTO81Ø

1ØØØ   PRINT"cdown YOU MAY TAKE 1, 2, 3 OR 4 SUCCESSIVE"
1Ø1Ø   PRINT"cdown READINGS, WHICH WILL BE STORED"
1Ø2Ø   PRINT"cdown AS WELL AS BEING DISPLAYED."
1Ø3Ø   PRINT"cdown WHEN YOU ARE READY TO START,"
1Ø4Ø   PRINT"cdown PRESS ONE OF THESE NUMBERS."
1Ø5Ø   GETB$:F=VAL(B$):IF F<>1 AND F<>2 AND F<>3 AND F<>4 THEN 1Ø5Ø
1Ø6Ø   PRINT"cdown "I'M READY. LET IT GO !"
1Ø7Ø   RETURN

3ØØØ   REM  DISPLAY ROUTINE IN MACHINE CODE
3Ø1Ø   FOR I=634 TO 729: READ X: POKE I,X:NEXT
3Ø12   DATA 169,13,133,252,169,178,133,253,169,13Ø,133,254,2Ø8,27
3Ø14   DATA 169,12,133,252,169,186,133,253,169,13Ø,133,254,2Ø8,13
3Ø2Ø        REM  DETERMINE STARTING LOCATION FOR DISPLAY
3Ø22   DATA 165,251,1Ø,1Ø,1Ø,1Ø5,24,133,253,169,129,133,254
3Ø25   REM DISPLAY REQUIRED BITS FOR EACH DIGIT
3Ø3Ø   DATA 162,Ø,165,252,1Ø,1Ø,1Ø,168,185,32,3,157,232,3,2ØØ,232,224,8,2Ø8,244
3Ø4Ø   DATA 16Ø,223,162,255,232,224,8,2Ø8,1,96,169,8,133,251,24,152,1Ø5,32,168
3Ø5Ø   DATA 169,16Ø,2ØØ,3Ø,232,3,144,2,145,253,198,251,24Ø,227,2Ø8,242

5ØØØ   REM BITS TABLE
5Ø1Ø       FOR I=8ØØ TO995:READ X:POKE I,X:NEXT
5Ø2Ø   DATA 124,68,68,68,68,68,124,Ø,8,8,8,8,8,8,8,Ø
5Ø3Ø   DATA 124,68,4,4,124,64,124,Ø,124,4,4,124,4,4,124,Ø
5Ø4Ø   DATA 64,64,64,72,124,8,8,Ø,124,64,64,124,4,4,124,Ø
5Ø5Ø   DATA 124,64,64,124,68,68,124,Ø
5Ø55   DATA 124,4,4,4,4,4,4,Ø
5Ø6Ø   DATA 124,68,68,124,68,68,124,Ø,124,68,68,124,4,4,4,Ø
5Ø7Ø   DATAØ,Ø,Ø,Ø,Ø,Ø,16,Ø,Ø,Ø,Ø,124,Ø,Ø,Ø,Ø
5Ø8Ø   DATA Ø,Ø,6Ø,32,6Ø,4,6Ø,Ø,Ø,Ø,254,146,146,146,146,Ø
5Ø9Ø   DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
```

```
6ØØØ  REM TIMER ROUTINE
6Ø1Ø  DATA 12Ø,16Ø,Ø,173,79,232,133,254,173,79,232,197,254,24Ø,249,133,254
6Ø2Ø  DATA 169,Ø,133,252,133,253,173,79,232,197,254,24Ø,22
6Ø3Ø  DATA 133,254,165,252,153,232,3,165,253,153,233,3,198,251,24Ø,4,2ØØ,2ØØ
6Ø4Ø  DATA 2Ø8,223,88,96,24,165,252,1Ø5,1,133,252,165,253,1Ø5,Ø,133,253,176,7
6Ø5Ø  DATA 162,13,2Ø2,2Ø8,253,24Ø,2Ø5,88,96
6Ø6Ø  RETURN
```

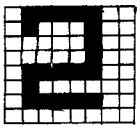
SUB-ROUTINE FOR DIGIT DISPLAY

Each digit is printed on the screen using a 7 x 5 dot-matrix format. This is "encased" in an 8 x 8 box, which allows for three columns to be used as spacers between adjacent digits. With a screen width of 40, five digits may be displayed in one line (or four and a decimal point, or three and a decimal point and minus sign).

Example:

Since each digit can be considered as eight rows and since each row can be stored by a single byte (one bit per column), this requires 8 bytes to store each digit used. The whole alpha-numeric system can thus be contained in about two pages of memory. Each row is represented by the decimal equivalent of the bits that have to be turned ON.

Example:



BITS	Decimal Equivalent
0 1 1 1 1 1 0 0	124
0 1 0 0 0 1 0 0	68
0 0 0 0 0 1 0 0	4
0 0 0 0 0 1 0 0	4
0 1 1 1 1 1 0 0	124
0 1 0 0 0 0 0 0	64
0 1 1 1 1 1 0 0	124
0 0 0 0 0 0 0 0	0

The following table gives the data bytes for each character used in this program.

124,68,68,68,68,68,124,0	0	8,8,8,8,8,8,8,0	1
124,68,4,4,124,64,124,0	2	124,4,4,124,4,4,124,0	3
64,64,64,72,124,8,8,0	4	124,64,64,124,4,4,124,0	5
124,64,64,124,68,68,124,0	6	124,4,4,4,4,4,4,0	7
124,68,68,124,68,68,124,0	8	124,68,68,124,4,4,4,0	9
0,0,0,0,0,0,16,0	. full stop	0,0,0,124,0,0,0,0	- minus
0,0,254,146,146,146,146,0	m m	0,0,60,32,60,4,60,0	s s
0,0,0,0,0,0,0,0	blank		

MACHINE CODE SUBROUTINE TO DISPLAY THE DIGITS

There are five digit positions and a numbe from 0 to 4 is POKED into DIGITPOSITION (location 251) prior to the SYS command. The subroutine uses this to calculate the screen position of the top left-hand corne of the digit to be displayed. The index Y is used to move the position of each dot i the matrix relative to this starting position. The digit to be displayed is POKED into DIGITVAL (location 252), from where the subroutine determines the position of the BITS by accessing the eigh bytes stored in the table (in locations 80 to 919). A white "blank" is stored at the correct screen location for each logic 1 bit. A logic 0 bit causes a branch so tha that screen location is left clear. Clearing the screen prior to a display is performed by BASIC; without this the digit could over-write each other.

The letter m and s (for s, m s^{-1} and m s^{-2}) are entered by similar, less complex routines, which have not been described separately. The -1 and -2 of this display is handled by BASIC.

RAM address	Decimal code	Mnemonic	Comment
662	165	LDA DIGITPOSITION	
663	251		
664	10	ASL	Multiply by 8 to fix
665	10	ASL	screen position
666	10	ASL	
667	105	ADC # 24	Add 24 to give initial
668	24		screen location of
669	133	STA SCNPOSLO	33048(LO 24, HI 129)
670	253		Keep screen position in
671	169	LDA # 129	zero page location for
672	129		indexed addressing
673	133	STA SCNPOSHI	
674	254		
675	162	LDY # 0	Clear index Y

728	208		BNE	8 bits not done, do next bit
729	242			

USER PORT TIMER ROUTINE (MACHINE CODE)

RAM Address	Decimal code	Mnemonic		Comment
920	120		SEI	To prevent PET interrupt from influencing timing loops
921	160		LDY # 0	Point to COUNTSTORE
922	0			
923	173		LDA VIAORA	User Port Input (Dec 59471)
924	79			
925	232			
926	133		STA STATE	Keep present state of User Port
927	254			
928	173	WAIT	LDA VIAORA	Wait till User Port changes state
929	79			
930	232			
931	197		CMP STATE	
932	254			
933	240		BEQ WAIT	
934	249			
935	133		STA STATE	User Port has changed, keep
936	254			new state of User Port
937	169	NEWCOUNT	LDA # 0	Clear temporary counters
938	0			
939	133		STA COUNTLO	
940	252			
941	133		STA COUNTHI	
942	253			
943	173	LOOP	LDA VIAORA	Check User Port for change
944	79			of state
945	232			
946	197		CMP STATE	
947	254			
948	240		BEQ COUNTON	If not, carry on timing
949	22			
950	133		STA STATE	Count has finished; keep new
951	254			User Port state
952	165		LDA COUNTLO	Put final count into permanent
953	252			store, for access by BASIC
954	153		STA CNTSTRLO,Y	
955	232			
956	3			
957	165		LDA COUNTHI	
958	253			
959	153		STA CNTSTRHI,Y	
960	233			
961	3			

```
962   198            DEC LOOPCHECK      LOOPCHECK poked in by BASIC to
963   251                               determine no. of loops reqd.
964   240            BEQ FINISH
965     4
966   200            INY                Update pointer
967   200            INY
968   208            BNE NEWCOUNT
969   223
970    88  FINISH     CLI
971    96             RTS
972    24  COUNTON    CLC
973   165             LDA COUNTLO        Add 1 (i.e. 100μ s) to counter
974   252
975   105             ADC ## 1
976     1
977   133             STA COUNTLO
978   252
979   165             LDA COUNTHI
980   253
981   105             ADC ## 0
982     0
983   133             STA COUNTHI
984   253
985   176             BCS ABORT          Overflow if time interval exceeds
986     7                                6.5535 s
987   162             LDX ## 13          D elay to 100μ s
988    13
989   202  DELAY      DEX
990   208             BNE DELAY
991   253
992   240             BEQ LOOP
993   205
994    88  ABORT      CLI
995    96             RTS
```

Sorting by Insertion & Chaining

L J Slow, Bradford Grammar School

This method requires only three passes through the array (A), together with some passing along short chains; thus the sort-time is approximately linearly dependent upon the number of elements (N+1).

The value of each element in A is transformed into an integer (E) in the range 0 to N, inclusive. This integer depends on the relative size of the value in A and is the position in the array B% where the element number of the current value should be stored. However, if this position is already occupied then a supplementary array C% is used to begin a chain of element numbers corresponding to those valued in A that give rise to the same E. Each link in the chain holds the element number of the next link; the last link holds the value -1. Links in each chain are positioned so that the corresponding values in A are in ascending order. At the end of the INSERTION and CHAINING loop, B% will contain the first links in each chain with -1's remaining where no chains begin, and C% will contain all the chains interleaved with each other and residual -1's.

This method is at its most efficient when the values to be sorted are in a uniform distribution. However, if the values follow a normal distribution, a function that maps this into a uniform distribution could be considered, but longer calculation time could offset the advantages of using a non-linear mapping function.

The subroutine listed below occupies 530 bytes after the REM's are deleted. It was tested with numbers in a uniform distribution generated by the RND function. Increasing N by 10 had the effect of increasing the sort-time by 1.0 to 1.2 secs. and the memory used by only 140 bytes. If the original list contains only integers then the arrays A and D can be replaced by A% and D% reducing this increase in running size to 80 bytes. New ROM's do not limit array size and a sort of 2000 numbers took 203 secs. on a 32K machine.

VARIABLE DICTIONARY

A	Array to be sorted.
B%	Array of first links of chains.
B	A particular element of B%.
C%	Array of all other chain links.
C	A particular element of C%.
D	Duplicate array of A.
E	Transformed element of A.
F	Mapping factor.
I,J	Counts.
MAX	Maximum value of A.
MIN	Minimum value of A.

All the arrays have N+1 elements.

LINE NUMBER EXPLANATION

50220	The current element number is inserted at the point E in the array B%, unless a chain already starts here.
50230	If a chain has been begun it then has to be decided if the first link in the chain should be displaced. This is done if the current value in A is less than that corresponding to the first link, in which case the first link becomes the second in the chain.
50240	However, if the first link is in the correct position, control passes to the next link. If there are no more links, the current element number is placed at the end of the chain.
50250	If there are more links, reference is made to the array A to decide whether the current element number should displace the next link, this being done if appropriate.
50260	If the current element number is still unplaced, control passes further along the chain.

EXAMPLE DRY RUN

Suppose that the array A has 51 elements (N=50), with MAX=70.0 and MIN=-30.0, F=0.5.

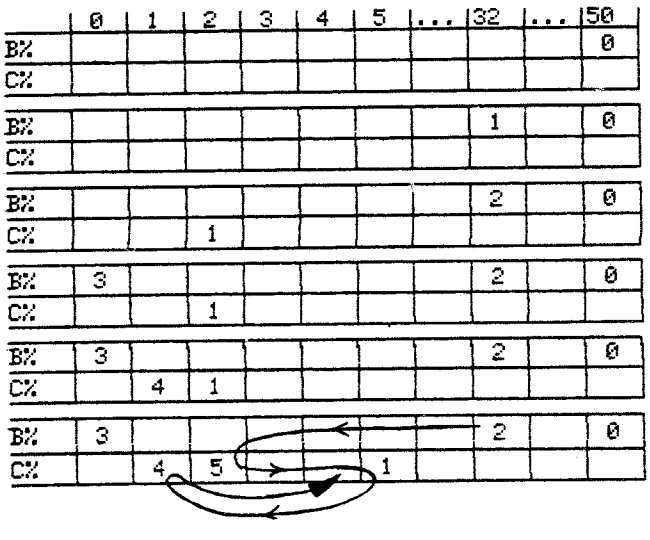
If the first six values in A are:

70.0, 35.5, 35.1, -30.0, 35.8, 35.3, ...,
0 1 2 3 4 5

then the corresponding E's are:

50, 32, 32, 0, 32, 32, ...,

The following shows how these values of E are used to insert and chain element numbers in the arrays B% and C% step by step. For the sake of clarity, -1's are omitted.



Note how the element numbers are used to point to the next link in the chain;
2→5→1→4 and 4 is the current end of this chain.
(35.1→35.3→35.5→35.8)

```
10 REM  TEST PROGRAM FOR SUBROUTINE.
20 PRINT"HOW MANY NUMBERS";:INPUT M:PRINT
30 IF(M<>INT(M))OR(M<2)GOTO 20
40 N=M-1:IF N<256 GOTO 70
50 PRINT"**WARNING,FOR OLD ROMS THE**"
60 PRINT"********LIMIT IS 256********"
70 DIM A(N)
80 FOR I=0 TO N
90 A(I)=(RND(TI)-.5)*1000:PRINTA(I)
100 NEXT
110 T=TI
120 GOSUB 50000
130 T=TI-T:T=INT(T*5/3+.5)/100
140 FOR I=1 TO N
150 IF A(I)>=A(I-1) GOTO 170
160 PRINT"ERROR":STOP
170 NEXT:PRINT
180 PRINT"FOR";M;"NUMBERS,TIME=";
190 PRINT T;"SECS."
200 PRINT"FREE MEMORY=";FRE(0);"BYTES."
205 PRINT:FORI=0TON:PRINTA(I):NEXT
210 END
215 REM   *   *   *   *   *   *   *   *
50000 REM  SUBROUTINE TO PLACE THE ELEMENTS OF A GIVEN ARRAY
50010 REM  IN ASCENDING ORDER BY INSERTION AND CHAINING
50020 REM
50030 REM  INPUT PARAMETERS: N & A(0),...,A(N)
50040 REM
50050 DIM B%(N),C%(N),D(N)
50070 REM  SET INITIAL VALUES AND FIND THE MAXIMUM AND MINIMUM VALUES OF A.
50080 MAX=A(0):MIN=A(0)
50090 FOR I=0 TO N
50100 B%(I)=-1:C%(I)=-1:D(I)=A(I)
50110 IF A(I)>MAX THEN MAX=A(I):GOTO 50130
50120 IF A(I)<MIN THEN MIN=A(I)
50130 NEXT
50140 IF MAX=MIN GOTO 50360
50150 REM
50160 REM  CALCULATE MAPPING FACTOR.
50170 F=INT(N*1000/(MAX-MIN))/1000
50180 REM
50190 REM  INSERTION AND CHAINING LOOP.
50200 FOR I=0 TO N
50210 E=INT((A(I)-MIN)*F):B=B%(E)
50220 IF B=-1 THEN B%(E)=I:GOTO 50270
50230 IF A(B)>A(I) THEN B%(E)=I:C%(I)=B:GOTO 50270
50240 C=C%(B):IF C=-1 THEN C%(B)=I:GOTO 50270
50250 IF A(C)>A(I) THEN C%(B)=I:C%(I)=C:GOTO 50270
50260 B=C:GOTO 50240
50270 NEXT
50280 REM
50290 REM  UNPACK CHAINS AND PUT ARRAY IN ORDER
50300 J=-1
50310 FOR I=0 TO N:B=B%(I)
50320 IF B=-1 GOTO 50350
50330 J=J+1:A(J)=D(B)
50340 B=C%(B):GOTO 50320
50350 NEXT
50360 RETURN
READY.
```

PROGRAMMABLE LINE FEED

One of the very useful features of the 3022 Tractor Feed Printers, which has been relatively unrecognised so far, is the ability to control the amount by which the paper is advanced for each line feed.

This programmable line feed function has many uses, dot by dot graph plotting is one application which has already been mentioned in CPUCN. But perhaps most users will be interested in the printer's ability to create attractive form layouts for tables of information.

The table illustrating the principle of sorting by insertion and chaining in Mr Slow's article was printed on the 3022 and the program listing appears below.

Line 20 opens a channel to special secondary address 6 which is used to control the programmable line feed. Line 100 sets the line feed spacing so that the vertical lines just touch one another. Line 109 decreases the spacing still further before advancing to print the next pair of boxes.

Line 98 resets the line feed to the normal value, which is preset when the printer is first turned on.

```
10 OPEN4,4
20 OPEN6,4,6
40 GOSUB100
50 R$=" ":S$="1":T$="0":X$=R$:Y$=R$:Z$=R$:GOSUB200
60 R$=" ":S$="2":T$="0":X$=R$:Y$="1":Z$=R$:GOSUB200
70 R$="3":S$="2":T$="0":X$=" ":Y$="1":Z$=X$:GOSUB200
80 R$="3":S$="2":T$="0":X$="4":Y$="1":Z$=" ":GOSUB200
90 R$="3":S$="2":T$="0":X$="4":Y$="5":Z$="1":GOSUB200
98 PRINT#6,CHR$(24)
99 END
100 PRINT#6,CHR$(18)
103 PRINT#4,"    | 0 | 1 | 2 | 3 | 4 | 5 |...|32 |...|50 |"
104 PRINT#4,"----+---+---+---+---+---+---+---+---+---+---+"
105 PRINT#4,"B%  |   |   |   |   |   |   |   |   |   | 0 |"
106 PRINT#4,"----+---+---+---+---+---+---+---+---+---+---+"
107 PRINT#4,"C%  |   |   |   |   |   |   |   |   |   |   |"
108 PRINT#4,"----+---+---+---+---+---+---+---+---+---+---+"
109 PRINT#6,CHR$(10):PRINT#4
110 RETURN
200 PRINT#6,CHR$(18)
204 PRINT#4,"----+---+---+---+---+---+---+---+---+---+---+"
205 PRINT#4,"B%  | "R$" |   |   |   |   |   | "S$" |   | "T$" |"
206 PRINT#4,"----+---+---+---+---+---+---+---+---+---+---+"
207 PRINT#4,"C%  |   | "X$" | "Y$" |   |   | "Z$" |   |   |   |"
208 PRINT#4,"----+---+---+---+---+---+---+---+---+---+---+"
209 PRINT#6,CHR$(10):PRINT#4:PRINT#6,CHR$(18)
210 RETURN
READY.
```

Re-Dimensioning Arrays

You all know what happens when you attempt to re-define an array that has already been defined. PET returns a ?REDIM'D ARRAY ERROR. But maybe you had a good reason to re-dimension. And now you must perform a CLR which clobbers all your variables, or else work around it. No longer! By manipulating some pointers down in zero page, arrays can be REDIM'D with no problem at all. Try the following example:-

```
100 DIM A$ (1000)
110 GOSUB 2000
120 DIM A$(2000)
130 GOSUB 2000
140 DIM A$(126)
150 END
200 REM
2000 POKE46,PEEK(44)
2010 POKE47,PEEK(45)
2020 Z9 = FRE(0)
2030 RETURN
```

The subroutine at 2000 "squeezes" the array out by making the End of Arrays Pointer equal to the Start of Arrays Pointer. PET now believes that there are no arrays of any name so DIMensioning is ok. The new array(s) is "built" in the same memory space.

Line 2000 forces a "garbage collection" so that any strings associated with Array A$ are thrown away. This wouldn't really be necessary with floating point or integer arrays since the values are stored in the array itself. With string arrays, only the string lengths and pointers to the strings are stored in the array. The strings lie elsewhere in RAM; in high memory if they were the result of a concatenation or INPUT from the keyboard, disk, etc. and directly in text if that is where they were defined (why store it twice?). This is also true for non-array type string variables. Of course, strings residing in text are not thrown away by garbage collection.

This trick can be played especially well when the sizes of your arrays are maintained in a disk file along with the file information.

Sometimes clearing all the arrays may not always be desirable. In that case, the order is which the arrays are defined becomes important. The 'permanent' arrays must be DIMensioned first, 'temporary' arrays last. However, if the value of the End of Arrays Pointer is stored immediately after defining the last 'permanent' array, the 'temporary' arrays can be squeezed out by POKing the End of Arrays Pointer with this value later on. For example:

```
100 DIM A(1000),B%(1500),C$(1450)
110 PL%=PEEK(46):PH%=PEEK(47)
120 REM
```

```
1000 INPUT#8,I%,J%,K%
1010 GOSUB 2100
1020 REM
2110 POKE46,PL%:POKE47,PH%
2120 DIMX(I%),Y%,(J%),Z$(K%)
```

The subroutine at 2100 would allow Arrays X, Y% and Z$ to be redimensioned any number of times without destroying Arrays A, B% and C$.

Dynamic Loading

Steve Punter of Mississauga has a note for those performing LOADs from within programs. If strings are defined in text and are to be passed between programs they must be placed in high memory before the LOAD is executed.

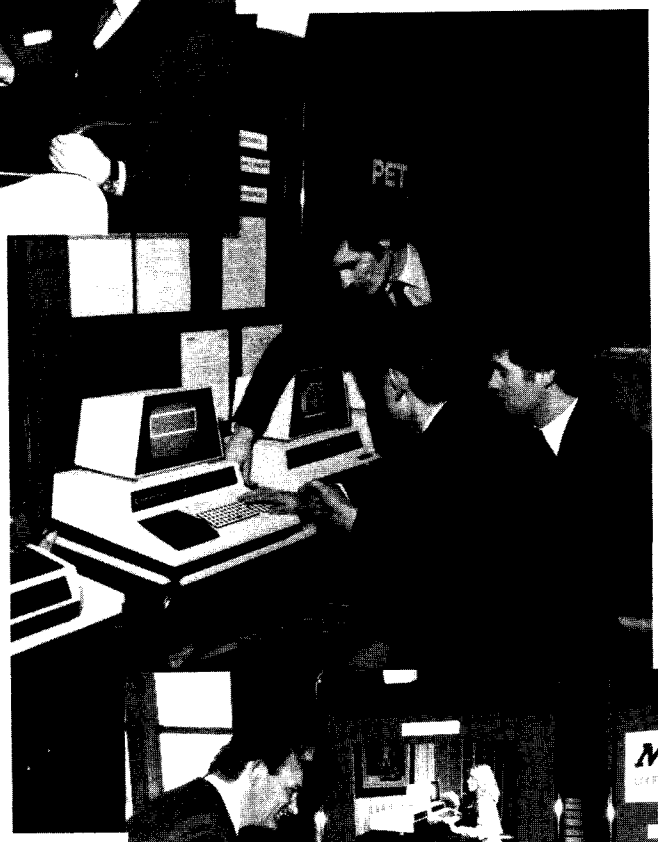
As mentioned earlier, a string variable is set up with only the length and a pointer to the location of the first character of that string. When strings are defined in part of a line of BASIC, this pointer points right into that part of text. A dynamic LOAD replaces that text with the new text and although the variable remains intact, the string itself is lost. In other words, the pointer does not change but what lies in that location and the locations following is not what it used to be. In fact, it could be virtually anything; BASIC command or keyword tokens, line numbers or even another (or part of another) string.

About the easiest way to avoid this is to define strings in text as a concatenation. For example:

```
50 SP$=""+"          "
60 NO$=""+"0123456789"
```

When a concatenation of any kind is performed, PET automatically rebuilds the string up in high RAM area thus protecting them from dynamic LOADs.

Medicom at G.P. Info 80: Dr. Dalton, author of diagnostic questionnaire program; Patrick Dixon, director; Dr. Emrys-Jones; author of the pharmacy dispensing package; a G.P. client; Dr. Meldrum, author of the practice monitor program.



Patrick Dixon illustrating a point to fellow Medicom director David Whitehead and Medicom clinical consultant Dr. Dalton.

A group of G.P.s examining the patient questionnaire program.

PETs in Medicine

The Commodore PET is already widely used in medicine - be it in general practice or in hospital laboratories. This was the clear message at GP INFO 80 and Computermarket. It is ideal because it is easy to use, small enough to fit into a GP's car, robust enough not to mind an occasional knock, and can easily be hitched up to medical equipment.

A company newly set up to assist the doctor considering a microcomputer system is MEDICOM, a subsidiary of Adda Computers. Medicom's director, Patrick Dixon, commented: "Since our launch at GP INFO 80 last month, and our appearance at Computermarket, there has been enormous interest by doctors, press and medical establishments. We will be appointing official dealerships of some of our products in the near future".

Most of the packages offered by Medicom have come initially from doctors themselves, who hand over their programs to be finished to commercial standards, documented, marketted and supported. Doctors in turn receive a royalty on each program sold. "Most doctors are delighted at the prospect of seeing a return for their investment, but don't want to give up medicine to run a business. Our arrangement gives them the best of both worlds - revenue with the minimum of inconvenience".

The first product, PRACTICE MONITOR, was conceived by Dr. Meldrum, an Ipswich GP, and has been in operation in his practice for almost six months. The set of three programs is designed to give a comprehensive breakdown of practice visits: which patients are seen by which doctors and why? This can be used, for example, to give the health visitor a list of 0-5 year olds with ear infections, seen in the last four weeks. It can be used to see whether Mrs. Jones - an elderly diabetic - saw any of the partners last month. Other uses include monitoring the GP trainee, keeping track of procedures for claiming fees, night visits, epidemiological surveys etc. By providing the doctor with a comprehensive picture of what is going on in his practice, it also enables him to plan his resources to best advantage.

Will confidentiality be a problem? Patrick thinks not. "We have had discussions with the BMA on this point. PET is ideal because it gives the doctor complete control of the information which is stored. Any information which is disclosed to another doctor is entirely at his discretion. Stored details are also impossible to relate to any individual without using the doctor's own register".

Patrick believes strongly that the most useful programs are likely to continue to come from the first generation of PET users themselves. "Medicine is subtle and complex - it takes many years to become a doctor and many years afterwards to fully understand the complexities and important issues in general practice. Hospital medicine is also highly specialised and diverse. There is a saying which is that it is easier for a doctor to learn how to program than it is for a programmer to become a doctor! Our experience has already borne this out".

Patrick feels the IEEE bus is also an important selling point in favour of the PET for a hospital system. "The PET bus is ideal for plugging in analysers. Coupled with simple diagnostic routines, the PET then becomes a most versatile tool". Patrick added a final work of caution, "The medical market is new and fragile. Whilst there is a great interest and some excitement in a number of doctors at present, one or two mistakes by small companies could be disastrous". Patrick recommends rigorous minimum standards such as a full 24 hour call-out maintenance contract as part of a standard package, together with installation and training.

It's all happening at the Poly

The Polytechnic of North London has become a centre of expertise on microcomputers and their applications. Many departments are working with microcomputers and quite a few PETs can be seen around the place. The Poly also provides a home to the North London Hobby Computer Club, quite a few of its staff and students being involved in the organisation of the Club.

Though the Club only started about a year ago, it is probably by now the largest club of this sort in the country. Under the chairmanship of Robin Bradbeer, quite a personality in the PET world, the Club has acquired a substantial reputation. The Club's large membership allows it to operate a number of sections dealing with specific areas of interest. Each of these is almost a club in itself. Interests catered for include assembling a microcomputer from a kit of parts, microcomputers in business, programming, games and, of course, the by now famous PET Users' section which, under the enthusiastic leadership of Barry Miles, has established strong Transatlantic links with that wizard of the North American PET world, Jim Butterfield.

This summer the Club will be breaking new ground by acting as host to the first London Computer Fair which is being organised by the Association of London Computing Clubs. The Fair is being run on July 11th and 12th in the main theatre of

the Polytechnic of North London. Further details are available from Olenka Pierozynska, the Exhibition coordinator, on 01-607 2789, ext. 2445.

User Club members in and around London will probably find the London Computer Fair worth attending - as well as the usual exhibition stands a number of special activities are planned including a Seminar, Workshop and Bazaar.

Meanwhile back at the Poly itself, the Department of Electronic and Communications Engineering is running an extra-mural course on the 6502 processor. Aimed at the advanced user, the course modules include :

Introduction to the 650X family of microelectronic components
6502 hardware organisation
Introduction to machine code programming
Basic concepts of machine code programming
Assembler programming
650X applications

The course will be especially suitable for programmers and engineers working professionally with the PET.

The dates of the courses will be 12-13 May, 23-24 June, 15-16 Sept, 13-14 Oct.

Further details of this course, as well as membership details of the North London Hobby Computer Club, can be obtained from

Robin Bradbeer
Department of Electronic and Communications Engineering
The Polytechnic of North London
Holloway
LONDON
N7 8DB

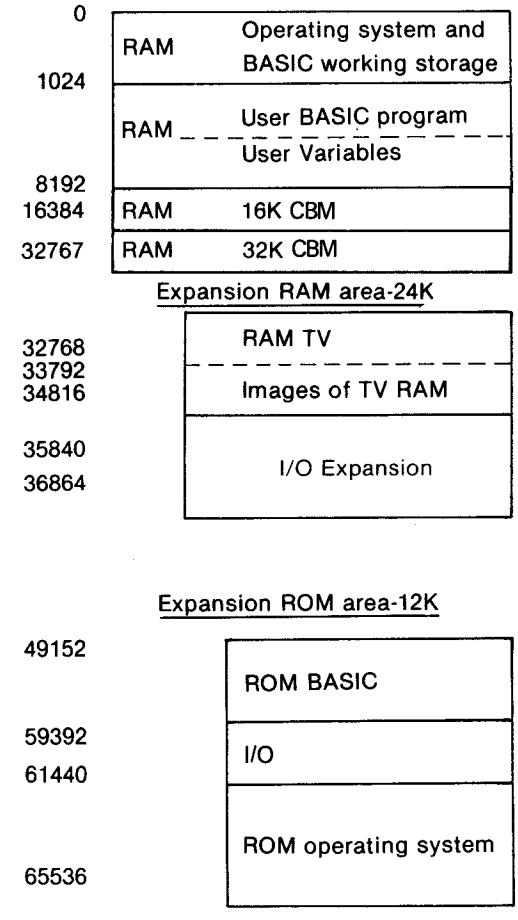
Beginning Machine Code

Paul Higginbottom

ADDRESSING

The 6502 uses 16 bits of memory to define the address of any byte of memory inside the PET. Consequently the highest address that PET can access, is the ammount represented by the largest possible sixteen digit binary number: 1111111111111111, which is 65535 in decimal. As a byte consists of eight bits and an address is 16 bits long then each address can be specified by two bytes, (apart from those special addresses 0 - 255, called zero page, which can be recognized by the 6502 with an address of only 8 bits). The rightmost eight bits or the lower half of the number is called the 'low byte' of the address and the upper eight bits or the second byte of the address is called the 'high byte'.

A single byte can range in value between 00000000 in binary (0 in decimal) to 11111111 in binary (255 in decimal). Memory is frequently referenced in 256 byte chunks (0-255). Each chunk is called a page of memory, and so the bytes with addresses 0-255 are called page zero (the 6502 has special zero page instructions), and the bytes 256-511 are called page one, etc. The PET uses pages 0,1,2,3 for it's own workspace; if we alter some of the values stored here(*), it is possible to make the PET operate in a different way. The table below outlines the way PET operating system uses its memory -



CBM memory map

* a detailed memory map for this area (0-1024 decimal) will appear in CPUCN 2.6

HEXADECIMAL NOTATION

This is the notation which is most frequently used by machine code programmers when referencing a number or address in an Assembly Language program.

Some assemblers make it possible to refer to addresses and numbers in decimal (base 10), binary (base 2), or even octal (base 8) as well as hexadecimal (or just 'hex'). Hexadecimal seems a bit difficult to grasp at first, but with practice it doesn't take long to master it.

If you look at a set of decimal (base 10) numbers, you will see that each digit in that number ranges between zero and a number equal to the base less one, i.e - 9. THIS IS TRUE OF ALL NUMBER BASES. Binary (base 2) numbers have digits ranging from zero to one (which is one less than the base). Similarly Hexadecimal numbers should have digits ranging from zero to fifteen, but as we do not have any single digit figures for the numbers ten to fifteen, so the first six letters of the alphabet are borrowed instead:-

Dec Hex

```
 0 - 0
 1 - 1
 2 - 2
 3 - 3
 4 - 4
 5 - 5
 6 - 6
 7 - 7
 8 - 8
 9 - 9
10 - A
11 - B
12 - C
13 - D
14 - E
15 - F
16 - 10
```

If that's confusing, let me try to put it another way:

```
   3   2  1  0
  10  10 10 10
--------------
1000 100 10 1
--------------
   4   5  6 9  = 4*1000+5*100+6*10+9
```

Example of how a base 10 (decimal number) is constructed.

```
   3   2  1  0
  16  16 16 16
--------------
4096 256 16 1
--------------
   1   1  D 9  = 1*4096+1*256+13*16+9
```

Example of how a base 16 (hexadecimal number) is constructed.

Therefore 4569(Base10) = 11D9(Base16)

The range for memory locations which the 6502 can address - 0 - 65535 (decimal) is 0 - FFFF in hexadecimal notation.

Usually hexadecimal numbers are prefixed with a dollar sign. This is to distinguish them from decimal numbers. Each address is only a four digit hexadecimal number ($0000 - $FFFF), and splitting the address into two byte format is easy as the leftmost two digits of a four digit hexadecimal address are the 'high byte', and the rightmost two digits are the 'low byte'.

This can be understood better if you think of binary again. Remember I said that the lower eight bits of an address constituted the 'low byte' and the upper eight bits were the 'high byte'. Well any 8 bit binary number can be uniquely represented by a two digit hex number. As the range of eight bits is 0 - 255, this is $00 to $FF in hex.

MACHINE CODE MONITOR

Let's now enter the machine code monitor. Mk 2 BASIC PET owners simply type:- SYS 1024 <RETURN>. Those with Mk 1 BASIC follow the instructions on the Commodore Machine Code Monitor tape available fromou local dealer. Your PET's screen will clear then come up with something like the display below though the actual numbers may be different.

```
B*
     PC  IRQ  SR AC XR YR SP
.;  0401 E62E 32 04 5E 00 F6
.
```

Don't worry about these numbers for the moment, but type in:-

.M 033A 0360 <RETURN>

You will see rows of 9 Hex numbers. The first 4 digit one is the address of the first byte of memory being shown in that row, and the other eight numbers are the actual contents of the memory locations beginning at that start address. Maybe the machine language monitor should have been named the Hex monitor although I realise that its name means that when you are using it, you are working at a 'machine level'.

I hope that now we have established the way that the PET accesses its memory, i.e as a sixteen bit number which is split in half as a lower and upper eight bits, we will soon be able to write our first machine code program!

In order to allow PET owners without assemblers to run the example programs associated with this "Beginning Machine Code" series of articles I will give the Hex listings as well as Assembly code versions. But since the I will be concentrating on what the machine code programs doing then it will be useful for everybody following these articles to study the programs in their most 'readable form', i.e. in assembly source code.

SAVING A MACHINE CODE PROGRAM

Let us assume that you know your program resides in the second cassette buffer which starts at decimal location 826 ($033A in hexadecimal). Having entered the monitor -

for resident monitor type:-

```
.S "FILENAME",01,033A,0400 <RETURN>
```

for tape based monitor type:-

```
.S 01,"FILENAME",033A,0400 <RETURN>
```

Where 'FILENAME' is the name you wish to call the program, '01' is the device number you wish to save the program on, '033A' is the start address, and '0400' is the address up to which you wish to save the contents of memory. To save on the disk (if you have new ROMs) you would need to specify the drive number in the filename as usual, and make the device number as '08'.

LET'S WRITE A MACHINE CODE PROGRAM

Now that I have described addressing, and how to create and save machine code programs, I think that it is about time we wrote one.

The 6502 has a central register known as an accumulator, which many of its machine code instructions act upon. It also has two other registers, one is called the X register, and the other (surprise, surprise) is called the Y register. These two registers can be used in the same way as the accumulator in most circumstances, nut their main function is that they can be used as indexes which means that they can be used as offsets to a particular address.

For example, if you wanted to draw a line of A's across the top of the screen in basic by using the POKE command, you could do this by typing in the following line:

```
PRINT"<CLR>" ; : FOR I = 0 TO 39 : POKE
32768 + I , 1 : NEXT <RETURN>
```

"1" is the POKE code for an 'A'

In this instance, 'I', which is a basic variable is being used as the offset from the main address which is 32768 ($8000 in hex). In machine code we could use the X register. For example:-

```
ADDR HEX CDES MNE OPER.  COMMENTS
033A A9 01    LDA #$01   ;LOAD ACCUMULATOR WITH
                         ;THE VALUE '1'. THE '#'
                         ;SYMBOL SIGNIFIES THAT YOU WISH
                         ;TO LOAD THE ACCUNMULATOR WITH
                         ;THE 'VALUE' 1, RATHER THAN THE
                         ;CONTENTS OF MEMORY $01

033C A2 00    LDX #$00   ;LOAD THE X REGISTER
                         ;WITH '0'.

033E 9D 00 80 STA $8000,X;STORE THE CONTENTS
                         ;OF THE ACCUMULATOR AT
                         ;HEX ADDRESS $8000
                         ;(WHICH IS 32768 IN
                         ;DECIMAL) OFFSET BY
                         ;OFFSET BY THE VALUE OF
                         ;X WHICH IS ZERO IN THE
                         ;FIRST CASE.

0340 E8       INX        ;INCREMENT THE X REG.

0341 E0 28    CPX #$28   ;COMPARE THE X REGISTER
                         ;WITH HEX $28 (DEC 40)

0343 D0 F8    BNE $033E  ;BRANCH IF IT'S NOT
                         ;EQUAL TO $28 BACK TO
                         ;HEX LOCATION $033E
0345 60       RTS        ;OTHERWISE RETURN FROM
                         ;SUBROUTINE I.E EXIT.
```

The Hex listing is as follows:

```
.M 033A 0342
.:  033A A9 01 A2 00 9D 00 80 E8 <RETURN>
.:  0342 E0 28 D0 F8 60 00 00 00 <RETURN>
```

Having either typed the program in the monitor or created it by using an assembler, you can run it by doing a SYS 826 <RETURN> command. I think I will leave this machine code program with you to ponder over and we will go over it next time.

THE PET AND A SPASTIC BOY'S POETRY

Christopher Nolan is a 14 year old spastic boy from Dublin who suffers from severe athetoid cerebral palsy. He cannot speak, and has almost no control over his movements. Three years ago teachers at a remedial school discovered that with the help of an anti-spastic drug and someone to hold his head, he had just enough control to pick out the keys on a typewriter. They found that he could not only read and spell but that he used words in a creative and chillingly apt way.

Taste of pity as people stare
Love, lots of love from mother.
Pills you find as lasting prayer
An irate person may possibly
Have faith instead of despair

Each word has been a formidable struggle against the spasms of his body. Sitting with a pointer attached to his headband, with his mother supporting his head, he directs the pointer towards the letter he wants. Sometimes his enthusiasm and frustration bring a violent spasm as he tries to select the appropriate key with his pointer.

Last December the "Sunday Times Magazine" printed a collection of Christopher's poetry which brought acclaim from University professors, critics and writers.

Subsequently, Philip Odor, a Research Fellow, at Edinburgh University devised a PET based system to help Christopher write. Christopher sits in front of PET's screen and a moving blob of light scans rows of letters and symbols. He presses a switch with his chin to select a row then an arrow hops along the selected row and he halts it by squeezing his knees together. The letter then appears in a sentence block on the screen. If he has misfired he simply stops the blob of light at the "cancel" signal and starts again. A word processor and stored dictionary program provides Christopher with further assistance. The PET floppy disk and printer was lent to Christopher for a few weeks by Software Serervices Development Ltd.

In February, the "Sunday Times Magazine" launched an appeal to raise £2200 to buy Christopher his PET outright. Response was overwhelming. The money needed for Christoper's system has been subscribed ten times over and will be used to provide a fund to help others with similar problems in communication. A special unit is being set up by the Central Remedial Clinic in Dublin, and Commodore are making available ten complete systems at half-price. Christopher writes of his PET:-

THE PET

My typewriter now is obsolete
It's told of my tongue-tied genetic dreams
Munificent mendicant hags
Spouted Cestus's loincloth
Cannot the awful poverty ever be halted momentarily
By a computerised friend

If you would like to help contribute towards the work of the unit please send donations to:

Christopher Nolan Appeal
Sunday Times
12 Coley Street
London WC99